N90-25569

A MODEL FOR A KNOWLEDGE-BASED SYSTEM's LIFE CYCLE

Peter A. Kiss
BDM International, Inc.
950 Explorer Boulevard
Huntsville, AL 35806

ABSTRACT

The American Institute of Aeronautics and Astronautics has initiated a Committee on Standards for Artificial Intelligence. Presented here are the initial efforts of one of the working groups of that committee. The purpose of this paper is to present a candidate model for the development life cycle of Knowledge Based Systems. The intent is for the model to be used by the Aerospace Community and eventually be evolved into a standard.

The model is rooted in the evolutionary model, borrows from the spiral model, and is embedded in the standard Waterfall model for software development. Its intent is to satisfy the development of both stand-alone and embedded KBSs. The phases of the life cycle are detailed as are and the review points that constitute the key milestones throughout the development process. The applicability and strengths of the model are discussed along with areas needing further development and refinement by the aerospace community.

1.0 INTRODUCTION

This paper presents a model for the life-cycle development of knowledge-based systems. The Artificial Intelligence Software Engineering (AISE) Model is an outgrowth of an effort by an AIAA committee on standards for AI. This committee was convened in early 1989 to explore the potential for developing various standards or guidelines for AI. Three working groups were formed to explore definitions and lexicon compilation, tools standardization feasibility, and development of life-cycle guidelines. The course of our approach is to develop candidate guidelines, disseminate to the community for feedback, and slowly evolve to standards as acceptance of the products grows. It is in that spirit that this paper presents the AISE model to the aerospace community for its feedback.

During the past ten years, the Knowledge-Based System (KBS) branch of Artificial Intelligence (AI) has matured considerably. Many small prototype systems have been successfully developed and implemented. Larger KBSs are much more complex and have been implemented at a slower rate. The organizations at the leading edge of using AI, ones that have been developing KBSs and applying them, are looking at the integration of KBSs into the mainstream of their computing environments. This is taking a more traditional total systems approach to AI, making the KBS an integral part, not a standalone tool. With the perspective of a systems approach comes the need for more rigorous development and integration methodologies. This need, coupled with general community's desire to control costs and schedules, is the impetus for the AISE model.

The objective of the AISE model is to provide a flexible framework for the development of a KBS (either standalone or integrated) with meaningful milestones and reviews that support the control of technical, cost, and schedule dimensions of a program. To achieve this objective, the model borrows the best attributes of the evolutionary software development model and some of the spiral model concepts and embeds them in the Waterfall model for software development.

2.0 SOFTWARE DEVELOPMENT MODELS

Several basic phases are inherent parts of any software (including AI) development program: Problem conceptualization/definition; system design; system development; testing; integration; and maintenance and enhancement. The

PRECEDING PAGE BLANK NOT FILMED

sequence in which these are carried out, the amount of emphasis/effort given each phase, and the controls associated with execution of the work combine to define a life-cycle model.

The Waterfall model, shown in Figure 1, is the most widely used in one variation or another. In the concept definition phase, studies and trades are conducted to define the system to be built. As a result, this phase culminates in a minimum of system requirements, top-level design specifications, and an operational concept. Next, a Preliminary Design Phase fleshes out the specifications and top-level design. Interfaces and data bases are specified, critical methods (such as special algorithms) are addressed, and test plans are conceived. The Preliminary Design is followed by a Detailed Design phase that finalizes the design and specifications. Simulations and prototyping are used to test the design, and test plans and operations

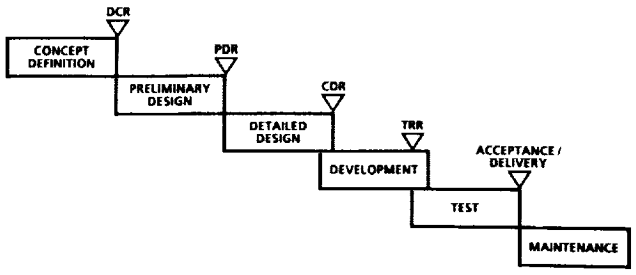

Figure 1. Standard Software Development Life-cycle

manuals are developed. Once design is complete, the software is coded/developed according to it and the specifications. As the software components are developed, they are tested and hierarchically validated and integrated to form the overall system. Once the system is accepted by its users, there is usually a long life of maintenance and upgrades during its operation.

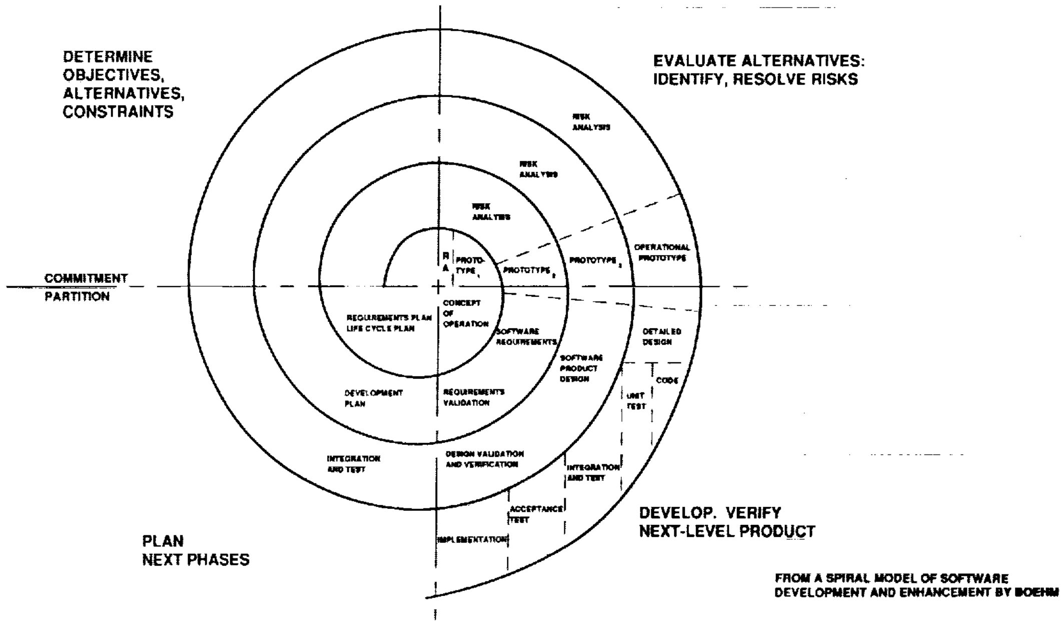

Figure 2. Spiral Model of the Software Process

The Spiral model, developed at TRW and shown in Figure 2, follows a different sequence. Once a problem is conceived, a series of prototypes is used to address the areas of highest risk in order of difficulty. Once all the parts of the system are well understood and the prototypes have developed a preliminary design, this model picks up the back phases of the Waterfall model to finish the product. A key characteristic of the Spiral model is the non-uniform maturation of system parts.

Another methodology for software development, one often used for AI, is the Evolutionary Model. Under this model, software is developed and tested incrementally for most of its life cycle.

Figure 3 provides a comparison of the models discussed. Given are the most appropriate situations for the application of each of the models, along with their strengths and weaknesses. If we examine the chart in light of some key characteristics of an aerospace KBS development, we are lead to the conclusion that a hybrid model is needed. Three characteristics of KBS development are essential for aerospace applications: 1) There is usually uncertainty in the scope of the problem and its appropriate solution; 2) the knowledge engineering process is inherently an evolutionary process; and 3) projects tend to have tight cost and schedule budgets. These three items point to a model that has flexibility and is evolutionary in nature while at the same time has a firm structure to control the development process. The Artificial Intelligence Software (AISE) model is designed to meet these needs.

MODELS	APPLICABILITY	STRENGTH	WEAKNESS
WATERFALL (SPECIFICATION DRIVEN)	· LARGE SCALE DEVELOPMENT · WELL DEFINED PROBLEMS · CONSTRAINED RESOURCES · GOVERNMENT REQUIREMENT	· RIGOROUS STRUCTURE · WORKS TO CONSTRAINTS · GOOD DEVELOPMENT VISIBILITY	· DIFFICULT TO CHANGE · UNIFORM PROGRES OF ALL COMPONENTS · DOES NOT ACCOMODATE EVOLUTIONARY DEVELOPMENT
SPIRAL (RISK DRIVEN)	· MEDIUM SIZE DEVELOPMENT · KNOWN RISKY AREAS · UNCONSTRAINED RESOURCES	· ACCOMODATES NON UNIFORM DEVELOPMENT · CONCENTRATE ON CRITICAL COMPONENTS · ADAPTIVE TO OTHER MODELS	· LIMITED COST AND SCHEDULE CONTROLS · LIMITED DEVELOPMENT OF MILESTONES AND REVIEWS · LIMITED SPECIFICATION AND DOCUMENTATION DEVELOPMENT
EVOLUTIONARY (PROTOTYPE DRIVEN)	· SMALL TO MEDIUM SIZE · ILL DEFINED PROBLEMS · UNCONSTRINED RESOURCES	· INCREMENTAL BUILD · EASY TO CHANGE DIRECTION	· LIMITED COST AND SCHEDULE CONTROLS · LIMITED CONTROL OF REQUIREMENTS · DIFFICULTY SCALING UP · NO VISIBILITY INTO PROCESS

Figure 3. Software Development Models

3.0 ARTIFICIAL INTELLIGENCE SOFTWARE ENGINEERING (AISE) MODEL

The AISE model, shown in Figure 4, focuses on the KBS element of a system as an area of high risk. It drives the development to be at the same level of maturity for its components at each major milestone, thus providing for process control.

The AISE phases and their relations to each other are shown in Figure 4. In the following sections, we discuss the objectives, activities, and results of each phase.

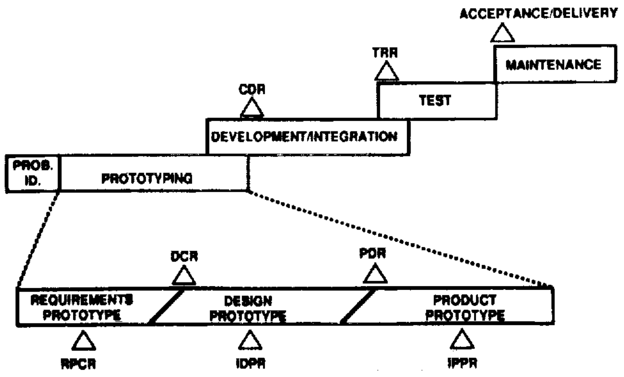

Figure 4. The AI Software Engineering (AISE) model

3.1 Problem Identification

Objectives: Analyze and define problem elements that are suitable for KBS solution.

Activities:

1. Isolate problem areas that are potentially suitable for KBS solution

2. Perform trades to determine whether KBS is the best solution compared to other techniques

3. Perform cost/benefit analysis

4. Draft development plans, including key participants needed

Results/Products: A well defined and justified KBS application with a plan for its development

3.2 Prototyping

Objective: Develop a full-capability prototype of the KBS element along with a detailed design for its target implementation

Activities: A series of three prototyping iterations and six reviews

1. Evolve a prototype to a full knowledge set
2. Test prototype during development
3. Develop documentation
4. Design target environment
5. Review prototyping progress

Results/Products: A fully developed product prototype of the KBS, the design for its target environment, and the associated support documentation.

The prototyping phase is the most critical in building a KBS, and accordingly it is the heart of the AISE model. The content and control of the work done in this phase will determine the success of the system being built. Figures 5, 6, and 7 show details of the review milestones associated with each of

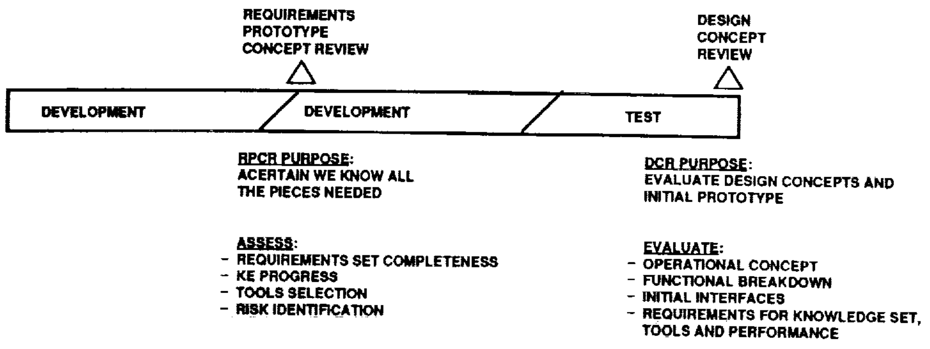
REQUIREMENTS PROTOTYPE CONCEPT REVIEW
DESIGN CONCEPT REVIEW
DEVELOPMENT
DEVELOPMENT
TEST
RPCR PURPOSE: ACERTAIN WE KNOW ALL THE PIECES NEEDED
ASSESS:
- REQUIREMENTS SET COMPLETENESS
- KE PROGRESS
- TOOLS SELECTION
- RISK IDENTIFICATION
DCR PURPOSE: EVALUATE DESIGN CONCEPTS AND INITIAL PROTOTYPE
EVALUATE:
- OPERATIONAL CONCEPT
- FUNCTIONAL BREAKDOWN
- INITIAL INTERFACES
- REQUIREMENTS FOR KNOWLEDGE SET, TOOLS AND PERFORMANCE

Figure 5. Requirements Prototype

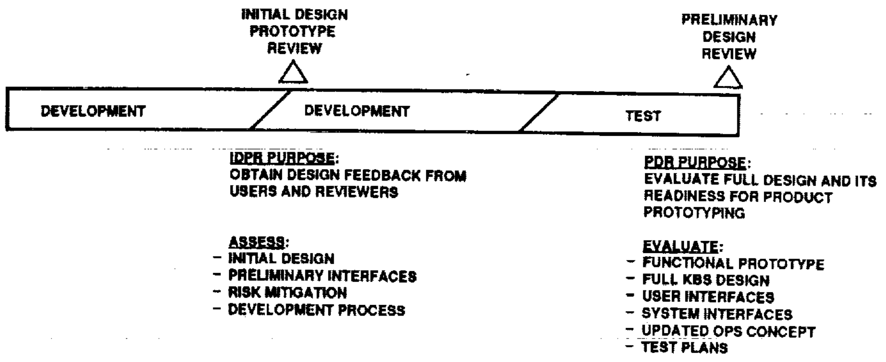

Figure 6. Design Prototype

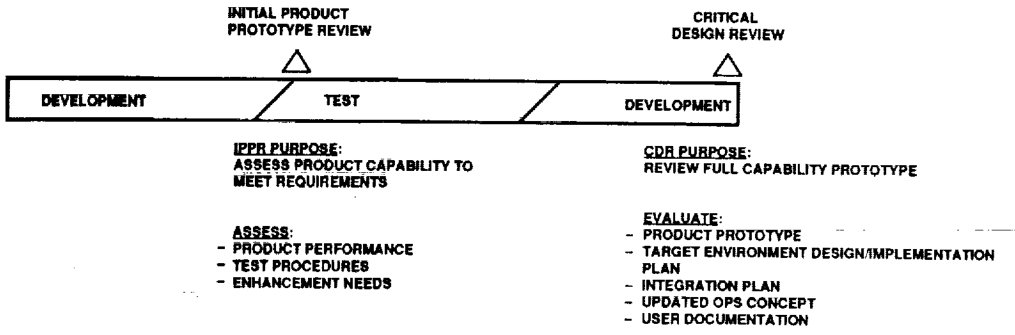

Figure 7. Product Prototype

the prototyping stages and the contents expected at each review.

3.3 Development/Integration:

Objective: Embed the KBS into its target environment.

Activities:

1. Port KBS to intended host environment (and, if applicable, language)

2. Integrate with system components that are external to the KBS through interfaces (I/F)

3. Implement integrated user I/F

4. Develop documentation

Results/Products: An integrated KBS in its target environment

3.4 Test and Evaluation

Objective: To ensure that the overall system works according to specifications and meets its requirements

Activities:

1. Perform hierarchical tests with greater levels of integration

2. Perform regression tests to check against standalone KBS prototype results

3. Evaluate overall system performance

4. Validate that the system meets requirements

5. Perform acceptance testing

Results/Product: Completed system ready for delivery to user

3.5 Operations and Maintenance:

Objective: Apply system to its intended use

Activities:

1. Routine operation of system

2. Debugging as required

3. Enhancements as the needs come up

Result/Product: A gracefully maturing system

The life-cycle of the AISE model has been planned to be compatible with the Waterfall model. This was done deliberately since many aerospace programs are mandated to use a variant of the Waterfall model (many are requested to use the 2167A standard). Figure 8 shows how the AISE model folds into the Waterfall. The review milestones align precisely with the completion of the prototyping phases and the two merge during the development phase.

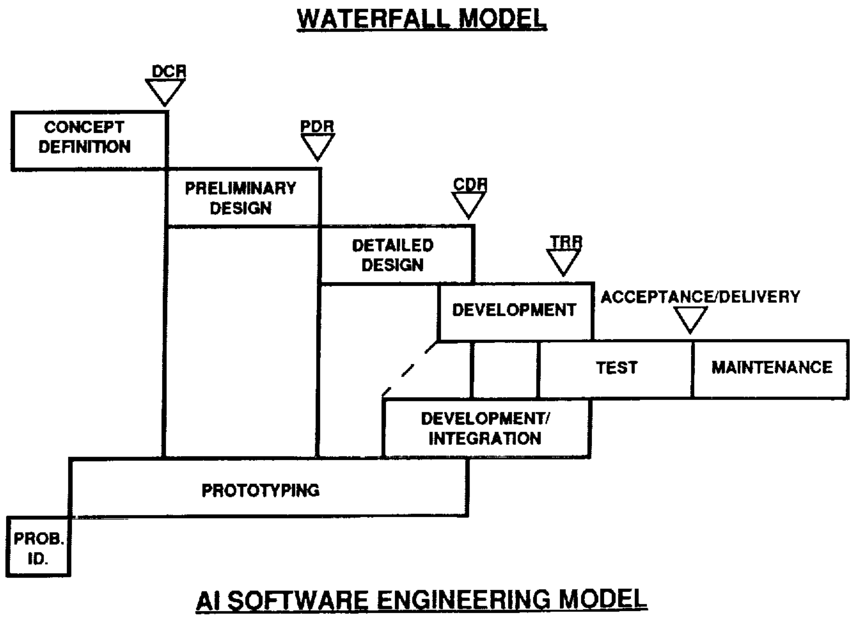

Figure 8. Integrated/Embedded Methodology

4.0 CONCLUSIONS

In order to integrate KBSs into the mainstream of software development and aerospace applications, a more rigorous development methodology is needed. The most popular software development models have been examined for their applicability and characteristics. A new hybrid, the AISE model, is proposed for KBS development. The AISE model provides flexibility up front for evolution of a knowledge base. At the same time, it provides visibility into development through a series of reviews. One of the features of the AISE model is the uniform (at milestones) development of all components of the KBS. This uniformity allows for meaningful development of requirements and specifications for the whole system, which in turn provides the mechanisms for technical, cost, and schedule controls. Finally, the AISE model can be neatly merged with the Waterfall model making the AISE model applicable and compliant with most Government software acquisition requirements.

5.0 ACKNOWLEDGEMENTS

This work is an outgrowth of the AIAA committee on standards for AI. The continued support of the following members is greatly appreciated: Robert Savely, Chuck Hall, Greg Swietek, Carol Russo, Guy Olney, Brad Allen, Mike Freeman, and Bruce Bullock. Also appreciated is the work of Nancy Seawell, with whom we formulated the early concepts of KBS development.

6.0 REFERENCES

1. B. Boehm, A Spiral Model for Software Development and Enhancement, ACM Sigsoft Software Engineering Notes, August 1988.

2. DOD STD 2167A and associated Software Specifications documents.

3. Dr. M. Freeman and P. Kiss, Issues in Management of Artificial Intelligence Based Projects, Fourth Conference on AI for Space Applications, November 1988.

4. F. P. Brooks, The Mythical Man-Month, Addison-Wesley, 1975

5. R. Pressman, Software Engineering, McGraw-Hill, 1987.

6. B. Boehm, Software Engineering Economics, Prentice-Hall, 1981.